

Mathematical Frameworks for Pricing in the Cloud: Revenue, Fairness, and Resource Allocations

Carlee Joe-Wong

Soumya Sen

Princeton University
{cjoe,soumyas}@princeton.edu

ABSTRACT

As more and more users begin to use the cloud for their computing needs, datacenter operators are increasingly pressed to effectively allocate their resources among these client users. Yet while much work has been done in this area, relatively little attention has been paid to studying perhaps the ultimate lever of resource allocation: pricing. Most data centers today charge users by “bundling” heterogeneous resources together in a fixed ratio and selling these bundles to their clients. But bundling masks the fact that different users require different combinations of resources (e.g., CPUs, memory, bandwidth) to process their jobs. The presence of multiple resources in fact allows an operator to offer many different types of pricing strategies, which may have different effects on its revenue. Moreover, to avoid user dissatisfaction, operators must consider the impact of their chosen prices on the fairness of the jobs processed for different users. In this paper, we develop an analytical framework that accounts for the fairness and revenue tradeoffs that arise in a datacenter’s multi-resource setting and the impact that different pricing plans can have on this tradeoff. We characterize the implications of different pricing plans on various fairness metrics and derive analytical limits on the operator’s fairness-revenue tradeoff. We then provide an algorithm to navigate this tradeoff and compare the tradeoff points for different pricing strategies on a data trace taken from a Google cluster.

Categories and Subject Descriptors

K.6.2 [Management of Computing and Information Systems]: Installation Management—*Pricing and resource allocation*

General Terms

Economics, Management

Keywords

Pricing, Cloud, Datacenters, Resource allocation

1. INTRODUCTION

1.1 Cloud Computing Resources

As Internet connectivity becomes ever more ubiquitous, cloud computing is becoming an increasingly popular alternative to traditional computing paradigms [26]. This dramatic increase in demand has led to several research challenges: how should a datacenter allocate multiple shared resources among users in a practical, scalable manner? The research implications of this question range from developing an efficient method of routing client requests to different datacenters, to allocating a certain amount of resources for processing each job as it arrives at a particular datacenter. In this work, we focus on an “infrastructure-as-a-service” view of the cloud, in which the resources considered are raw computing resources such as CPU time or memory.

When provisioning resources to users, datacenter operators must take into account the fact that multiple types of resources are being allocated. For instance, users’ jobs require both CPU time and RAM, which are non-substitutable resources: more RAM does not mitigate a need for CPU processing time, and neither does more processing time significantly lessen RAM requirements. This provisioning of multiple resources thus introduces new challenges to the datacenter operator’s resource allocation problem, which are only beginning to be studied [11]. In this work, we consider the consequences of multiple resource allocation on the prices offered to datacenter users.

1.2 Pricing Objectives

As datacenters become ever more popular, many position papers have argued for the need to research new pricing mechanisms that can accommodate the competing objectives of a datacenter operator [4, 26]. While the operator of course wishes to maximize its revenue, its pricing policy must also take into account the dynamic nature of user demand: as users submit jobs to the datacenter, the operator must be able to accommodate the influx of new jobs and balance the needs of incoming users with those of users already in queue. We thus introduce a fairness dimension: to avoid incurring user dissatisfaction, the datacenter operator should allocate its resources so that each user can process a fair, or equitable, number of jobs. Treating users fairly, however, can in itself impact revenue. In this work, we introduce an optimization framework that allows datacenter operators to balance the revenue and fairness of its resource allocation through setting prices for its computing resources.

The presence of multiple resources gives a datacenter op-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

erator flexibility in offering different types of pricing plans. In this work, we examine the implications of different pricing schemes on the achieved fairness and revenue. We consider three different pricing plans:

Bundled pricing: Bundled pricing assumes that operators group heterogeneous resources such as CPU time and memory together and then sell this resource bundle to users. For instance, Google’s Compute Engine sells virtual machines (VMs), each with a fixed CPU and memory capacity [7]. Basic pricing policies for Amazon’s EC2 similarly offer VM bundles [2]. Users are charged at a fixed unit rate per VM; they are free to make use of all or part of each VM resource. For instance, a user with computationally intensive jobs might not require the VM’s full RAM capacity, but will need to purchase its use anyway.

Resource pricing: Under resource pricing, an operator charges users separate unit prices for each resource that they use. Microsoft’s Windows Azure, for instance, charges separate unit rates for CPU time and memory [14]. Unlike bundled pricing, resource pricing does not charge users for resources they do not use. One can, however, view resource pricing as a form of bundled pricing, in which the bundles for each user are customized to exactly fit that user’s resource requirements. The price of each bundle is determined by the unit prices of the different resources.

Differentiated pricing: In differentiated pricing, users are allocated exactly as much of each resource as they require to process their jobs. Differentiated pricing differs from resource pricing, however, in that the operator can freely determine each user’s per-job cost, and is not constrained to charge users by unit resource prices. While differentiated pricing has not yet been introduced to the market, many believe that it will be necessary in the future [16].

In addition to the three pricing plans introduced above, one may take advantage of datacenters’ economies of scale by introducing **volume discounts** into a pricing policy; Amazon’s EC2 offers such a pricing plan, charging a lower unit rate for users who purchase a large quantity of resources [2]. While some works have investigated the effects of volume discounts, e.g., for contract users [3], none have done so in a multi-resource context. In this work, we compare bundled, resource, and differentiated pricing with volume discounts in a mathematical framework that takes into account user demand, operator revenue, and fairness across users.

While the pricing plans offered by specific operators have been analyzed and compared [18, 23], we use a general model to solve for an operator’s optimal prices. Our work is unique in accounting for both the pricing of multiple resources and the fairness of the resulting allocation. While other papers have accounted for fairness by using game theory to model user competition [19] or studying the strategy-proof property [20], these works do not explicitly deal with the presence of multiple, non-substitutable resources. Works that do consider multi-resource fairness, on the other hand, do not include user pricing. Instead, operators are assumed to allocate a certain number of jobs to users in a manner that is both efficient, in terms of a large number of processed jobs, and equitable, i.e., no user processes a disproportionately high or low number of jobs [6, 11, 22].

1.3 Job Arrivals and Deadlines

Dynamic pricing has been perhaps the most common approach taken in the literature on pricing for cloud or data-

center computing. Some papers have investigated revenue-maximizing prices [5, 27], while other models take into account such factors as job completion deadlines [24]. In another variant, users’ willingness to pay and job prices vary depending on a given job’s completion time [21]. For instance, operators may give different price quotes to users as a function of the completion time, and the user can select from these quotes [10]. Such a formulation may be used to maximize the long-term social surplus, defined in terms of user utilities over time [13]. However, most of these works do not take the presence of multiple resources into account. Other works have proposed dynamic auctions, an approach popular with grid computing [8, 17]. However, such measures, e.g., the spot pricing offered by Amazon’s EC2, have been shown to be ineffective in practice [25].

In our work, we suppose that the datacenter operator optimizes its prices over a finite time horizon. We divide time into discrete intervals, e.g., of one hour, and suppose that users are allocated the use of a given resource (e.g., CPU) for each particular interval.¹ We then have two possibilities: in the first, user demands could be the same during each time interval. The operator can then compute its prices in an offline optimization and simply offer those prices during each interval. In the second, more realistic, scenario, users’ demands change over time, and the operator must change its prices accordingly. Operators must then account for job deadlines, so that users’ jobs complete in a time acceptable to users. In the main body of this work, we consider the first scenario of optimization in a single time interval; Appendix A shows how our work may be extended to account for changes in user demand and job deadlines.

We begin developing our framework in Section 2 by examining users’ choices of how many jobs to submit, given a set of offered prices. Section 3 then discusses bundled, resource, and differentiated pricing in the context of users’ choices and derives expressions for the operator’s revenue. In Section 4, we introduce a family of metrics to measure the fairness under different pricing plans. We then characterize the fairness-revenue tradeoff with these metrics and derive algorithms for practically choosing prices at different points on the tradeoff. Finally, in Section 5 we perform simulations on a dataset from a Google cluster to compare the achieved fairness and efficiency for different pricing plans and a range of resource capacities and volume discounts. Section 6 concludes the paper.

2. USER OPTIMIZATION

Suppose that a datacenter operator has multiple clients (i.e., users), each of whom submits jobs to the datacenter. Each job requires a certain amount of datacenter resources, e.g., CPU time, memory, and bandwidth; we suppose, as is often the case in practice, that these resources are not substitutable. We index the resources with the variable i , and denote the amount of resource i required for each job by R_{ij} , where j indexes the user type. Different types of users are distinguished by the different resource requirements of their jobs; in an extreme case, each user might submit jobs with slightly different resource requirements and thus belong to a distinct type. Multiple users could also be grouped to-

¹If the time intervals are very short, this pricing is sometimes called *resource-as-a-service*, i.e., the selling of individual resources at very short time granularities [1].

gether into the same type, if their jobs have similar resource requirements that we can approximate as being the same. We fix the number of user types as n and the number of resources as m . If users are grouped together into a few types, then we could have $n \leq m$; otherwise, in general we have $n \geq m$ as there are many more user clients than types of resources.

When deciding how many jobs to submit, users take into account the prices set by the datacenter operator, which yield a cost per job for each user j . The structure of this pricing plan may vary; Section 3 introduces three different ways (bundled, resource, and differentiated pricing) in which the prices may be set. In all three pricing plans, each user has a fixed per-job cost, which we denote as r_j for user j . This cost accounts for both the resource prices and the volume discounts, which are taken to be separate functions of the amount of each resource required by the user. We parameterize the volume discount with a constant $\gamma \in (0, 1]$; if a user requires $R_{ij}x_j$ amount of resource i , where x_j denotes the number of jobs submitted by user j , then the user is charged according to the discounted quantity $R_{ij}^\gamma x_j^\gamma$. Thus, a larger γ corresponds to a smaller volume discount, and $\gamma = 1$ corresponds to no volume discount. Since the amount of resources required is an affine function of the user's demand, we let r_j denote the per-job cost with the volume discount and separately account for the discounted x_j^γ term.

Users derive utility from the number of jobs that the datacenter processes, but lose utility from the amount that their jobs cost.² We use x_j to denote the number of jobs processed for each user j , and use the function $U_j(x_j)$ to denote the utility from these jobs. Since a user's utility increases with the number of jobs processed, we follow the economic principle of diminishing marginal utility and assume that U_j is a strictly increasing, concave function, expressed in units of dollars. Subtracting the cost $r_j x_j^\gamma$ from the utility of processing x_j jobs, we find that each user j maximizes

$$U_j(x_j) - r_j x_j^\gamma. \quad (1)$$

For simplicity, we assume that U_j is continuously differentiable for $x_j > 0$. We suppose that the users have sufficient available cash to pay for any number of jobs x_j , so long as (1) is positive; for instance, enterprise users would meet this requirement. Users thus have no explicit budget constraint; they submit as many jobs as required to maximize (1) for a given set of prices.

An example family of utility functions are the so-called *isoelastic* functions

$$U_j(x) = \begin{cases} c_j(1 - \alpha_j)^{-1} x^{1-\alpha_j}, & \alpha_j \in (0, 1) \\ c_j \log(x) & \alpha_j = 1 \end{cases} \quad (2)$$

commonly used in economics [15]. In (2), α_j parameterizes the concavity of the utility function (i.e., quantifying the degree of diminishing marginal utility) and $c_j > 0$ is a positive constant that scales the utility level for user j . As α_j increases, U_j becomes more concave, and the user becomes more price-sensitive since the cost term $r_j x_j^\gamma$ term becomes more dominant.

We assume that the user receives zero utility if no jobs are processed; this assumption may be enforced by taking

²We assume throughout that the operator sets prices so that all submitted jobs can be processed with the resources available. "Submitted" and "processed" can thus be used interchangeably.

$U_j(0) = 0$. The maximizer x_j^* of (1) then satisfies

$$U_j'(x_j^*) = r_j \gamma x_j^{*\gamma-1}, \quad (3)$$

where x_j^* , a function of the per-job price r_j , denotes the utility-maximizing number of jobs.

We assume that the operator chooses a value of γ such that a solution to (3) exists, and that (1) has non-positive second derivative at this value of x_j^* , ensuring that the solution is indeed a maximum. Mathematically, the second derivative condition may be expressed as

$$U_j''(x_j^*) < \gamma(\gamma - 1)x_j^{*\gamma-2} r_j. \quad (4)$$

For instance, the isoelastic utility functions satisfy this condition if $\gamma > 1 - \alpha_j$. We can then show that user j 's demand x_j^* is decreasing in r_j . By differentiating (3), we find that this rate of decrease is

$$x_j^{*'}(r_j) = \frac{\gamma x_j^{*\gamma-1}}{U_j''(x_j^*) + \gamma(1 - \gamma)x_j^{*\gamma-2} r_j}, \quad (5)$$

which is negative if (4) holds.

3. PRICING DATACENTER JOBS

The presence of multiple datacenter resources allows the datacenter operator to offer many different pricing plans, each of which may be combined with volume discounts. In Section 3.1, we discuss the three pricing plans introduced in Section 1.2: bundled, resource, and differentiated pricing. We incorporate these prices in the user-optimization framework introduced in the previous section and then compare the optimal prices and resulting revenue in Section 3.2.

3.1 Pricing Strategies

We first consider **bundled pricing**, in which the operator groups different resources in a fixed ratio—for instance, 1 CPU and 2 GB of RAM might be grouped together as a bundled resource. The operator can then charge users a unit price per bundle of resources required for the user's jobs. We let b_i denote the amount of each resource $i = 1, \dots, m$ in each bundle, and let p denote the per-bundle price. Using the notation from Section 2 to describe users' utility functions and per-job resource requirements, we find that each user j requires

$$\mu_j = \max_i \left(\frac{R_{ij}}{b_i} \right) \quad (6)$$

bundles in order to complete one job. To calculate the per-job cost, we incorporate the volume discount introduced in the previous section, parameterized by $\gamma \in (0, 1]$. The per-job cost then becomes $r_j = \mu_j^\gamma p$. Moreover, the number of bundles available is $\min_i C_i/b_i$, where C_i denotes the capacity of resource i . Thus, the operator faces the resource constraint

$$\sum_{j=1}^n \max_i \left(\frac{R_{ij}}{b_i} \right) x_j^* (\mu_j^\gamma p) \leq \min_i \frac{C_i}{b_i}, \quad (7)$$

where $x_j^* (\mu_j^\gamma p)$ denotes the user's demand for jobs at price p , given the bundle (b_1, \dots, b_m) . The resulting revenue under bundled pricing, ρ_b , may be expressed as

$$\rho_b = p \sum_{j=1}^n (\mu_j x_j^* (\mu_j^\gamma p))^\gamma. \quad (8)$$

We next observe that ρ_b in (8) is a non-differentiable function of the amount of each resource i (i.e., b_i) in the bundle; thus, the operator will likely find it difficult to optimize over the b_i . Instead, it can fix these b_i and simply optimize over the unit price p . For instance, the b_i might be chosen as proportional to the resource capacities C_i , i.e., such that for some constant b , $b = C_i/b_i$ for each resource i . In this case, the user's bundle requirement (6) is proportional to her dominant share, which is defined as follows:

Definition 1. A user's **dominant share** is the maximum share of any resource allocated to the user. Thus, if a user receives $R_{ij}x_j^*$ amount of each resource i , her dominant share would be $\max_i R_{ij}x_j^*/b_i$, i.e., the bundle requirement (6) multiplied by b .

In Section 4.2, we show that users' dominant shares may be used to define the fairness of their utilities received.

While bundled pricing is convenient for the operator in the sense that only one price variable needs to be chosen, bundled pricing may lead to wasted capacity. Since different users' jobs have different resource requirements, the ratio of resources in each bundle will not in general match the ratio of per-job resource requirements for a user. Thus, some users may be forced to purchase excess capacity of some resources—for instance, a user with a per-job requirement of 2 CPU cores and 1 GB of RAM would need to purchase 4 CPU cores and 1 GB of RAM for each job if the operator offered bundles of 4 CPU cores and 1 GB of RAM. To avoid this wasted capacity, which could potentially be sold to another user with different resource requirements, an operator could offer **resource pricing**, in which the unit price of each resource is independently determined by the operator, and users are permitted to purchase exactly as much of each resource as they need to process their jobs.

In this pricing framework, we let p_i denote the unit price of each resource i , so that each user j 's per-job cost is $r_j = \sum_{i=1}^m p_i R_{ij}$. Thus, the operator's total revenue ρ_r is

$$\rho_r = \sum_{i=1}^m p_i \sum_{j=1}^n \left(R_{ij} x_j^* \left(\sum_{i=1}^m p_i R_{ij} \right) \right)^\gamma. \quad (9)$$

The resource constraints, which require that the operator be able to provide enough resources to each user, can then be expressed as

$$\sum_{j=1}^n R_{ij} x_j^* \leq C_i \quad (10)$$

for each resource i . The left-hand side corresponds to the sum of users' required amounts of resource i , while the right side is simply the capacity of resource i .

We next note that resource pricing may be viewed as a form of bundled pricing in which the ratio of resources in each bundle exactly matches that required by users. This relationship between bundled and resource pricing suggests that the operator can offer a third, even more customized type of pricing—**differentiated pricing**, in which the operator not only allows users to receive only as much of each resource as they need, but independently sets the prices of these customized bundles. Thus, the operator can set a different per-bundle price \bar{p}_j for each user j . We suppose that one bundle exactly equals the user's resource requirements for one job, so that the per-job cost $r_j = \bar{p}_j$. The operator's

revenue ρ_d under differentiated pricing is then

$$\rho_d = \sum_{j=1}^n \bar{p}_j x_j^* (\bar{p}_j)^\gamma, \quad (11)$$

and the resource constraints are exactly those under resource pricing (10).

3.2 Comparing Pricing Plans

In this section, we compare the operator's revenue under the three pricing plans introduced in the previous section. We suppose that for any of the three pricing plans, the operator chooses the prices so as to optimize a given objective function of users' per-job costs and demands, e.g., the revenue (8,9,11). In Section 4.2, we define the fairness of users' utilities, which form another family of possible objective functions.

We first give sufficient conditions under which the objective value under bundled pricing is at most that under resource pricing:

LEMMA 1. If R_{ij}/b_i for each user j is maximized by resource $i = k$, then the maximum objective function value with bundling does not exceed the maximum objective with unbundled resources.

PROOF. Suppose without loss of generality that $k = 1$. Each user's per-job bundle requirement is then R_{1j}/b_1 . Let the optimal price with bundling be denoted by p^b . Then if the operator charges $p_1 = p^b/b_1$ for resource 1 and nothing for any other resource, each user's per-job cost without bundling is $R_{1j}p^b/b_1$, which equals the cost with bundling. Each user's demand is solely determined by the price per job, so each user demands the same amount of jobs with and without bundling, and the resource constraints are satisfied. Since the objective function can be written entirely in terms of users' per-job costs, the value of the objective function is also unchanged. Thus, the maximum objective without bundling is at least as much as that with bundling. \square

Given a set of resource requirements, a bundle always exists that satisfies the requirements of Lemma 1—for instance, the operator might simply set b_1 to be extremely small. Such a scenario, however, is somewhat artificial. If an operator instead sets the ratio of resources in each bundle equal to the ratio of resource capacities, then Lemma 1 holds if each user has the same dominant resource, defined as follows:

Definition 2. A given user's **dominant resource** is the resource k which maximizes R_{ij}/C_i over all resources i .

We thus have the following corollary:

COROLLARY 1. Suppose that the ratio of resources in each bundle is equal to the ratio of resource capacities. Then if each user has the same dominant resource, the operator's maximum objective function value with bundling does not exceed that without bundling.

PROOF. If the amount of each resource i in the bundle is b_i , then there exists a constant B such that $Bb_i = C_i$ for each resource i . Thus, the resource k maximizing R_{ij}/C_i also maximizes R_{ij}/b_i , and Lemma 1 applies. \square

Intuitively, one might expect a result of this type, as the operator can choose only one price p under bundled pricing,

while it chooses $m \geq 1$ resource prices p_i under resource pricing. We note, however, that in general the operator's revenue under bundled pricing may not be less than that under resource pricing—under bundled pricing, users often are required to purchase excess capacity, and the resulting excess revenue may offset the operator's decreased flexibility in choosing prices. However, when comparing resource pricing and differentiated pricing, we obtain a stronger result:

PROPOSITION 1. *If an operator prices jobs by unit prices p_i for each resource i , its maximum objective function value does not exceed the optimal value under differentiated pricing. If the matrix of resource requirements \mathbf{R} has rank n , then the optimal objective function value under differentiated and resource pricing is the same.*

PROOF. Suppose that the optimal prices under resource pricing are p_i for each resource i . Let the differentiated prices $\bar{p}_j = \sum_{i=1}^m R_{ij} p_i$, so that the price per job does not change for any user. Then x_j^* , the user's demand for jobs, does not change either, as it depends solely on the per-job cost. The objective function value under these differentiated prices thus attains the maximum revenue under resource pricing. Moreover, since the resource constraints depend only on the number of jobs x_j^* , these differentiated prices are a feasible solution.

If the matrix R has rank n , then the resource prices may be set in such a way that the per-job cost for each user equals that under differentiated pricing. Thus, since the objective depends only on per-job costs, the optimal objective value under resource and differentiated pricing is the same. \square

In general, a datacenter operator will have several more clients than resources (i.e., $m \leq n$)—the number of datacenter resources will generally be a small number, e.g., two for CPU cores and memory. However, if we instead group users into a few distinct types, then the $m \times n$ matrix of resource requirements \mathbf{R} may have rank $n \leq m$.

4. FAIRNESS AND REVENUE

In this section, we examine a datacenter operator's two objectives: revenue and fairness. We first characterize the operator's revenue in Section 4.1, and then consider fairness in Section 4.2. Section 4.3 then constructs an optimization framework that takes into account both revenue and fairness. We derive bounds on the tradeoff between fairness and revenue and give an algorithm that finds the optimal prices for bundled, resource, and differentiated pricing at various points on this tradeoff curve.

4.1 Revenue and Resource Constraints

From the previous section, we see that the operator's revenue may be expressed as

$$\rho = \begin{cases} p \sum_{j=1}^n (\mu_j x_j^* (\mu_j^\gamma p))^\gamma \\ \sum_{i=1}^m p_i \sum_{j=1}^n (R_{ij} x_j^* (\sum_{i=1}^m p_i R_{ij}))^\gamma \\ \sum_{j=1}^n \bar{p}_j x_j^* (\bar{p}_j)^\gamma \end{cases} \quad (12)$$

for the bundled, resource, and differentiated pricing plans respectively. Should the operator wish to maximize its revenue, a natural approach would be to take the first derivative of the revenue with respect to each price variable and set it equal to zero. In this case, however, we can show that the

revenue is always decreasing in each price variable, assuming that the utility function U_j is not too concave compared to the cost $r_j x_j^\gamma$. Intuitively, each user j 's total utility should grow slowly enough with the prices so that the optimal demand x_j^* is well-defined, yet grows sublinearly so that the overall amount paid $r_j x_j^\gamma$ decreases with the prices bfp .

PROPOSITION 2. *Revenue ρ is decreasing in price if, for all users j ,*

$$0 < U_j''(x_j^*) + \gamma r_j x_j^{*\gamma-2} < \gamma^2 r_j x_j^{*\gamma-2}, \quad (13)$$

where r_j denotes the per-job cost for user j and is an affine function of the prices offered. For instance, the isoelastic utility functions (2) satisfy these criteria if, as assumed in Section 2, $\gamma > 1 - \alpha_j$.

PROOF. We take the first derivative of the revenue ρ with respect to a price variable p_k (which may correspond to any of the three pricing plans) to obtain

$$\frac{\partial \rho}{\partial p_k} = \sum_{j=1}^n \frac{\partial r_j}{\partial p_k} x_j^* (r_j)^\gamma + \sum_{j=1}^n r_j \frac{\partial r_j}{\partial p_k} \gamma x_j^{*\gamma-1} x_j^{*'} (r_j).$$

Since $x_j^{*'} < 0$, this derivative is negative if, for each user j ,

$$-x_j^{*'} > \frac{x_j^*}{\gamma r_j}.$$

We can then use (5) to obtain the equivalent condition

$$U_j'' > -\gamma r_j x_j^{*\gamma-2},$$

which is the first inequality in (13). The second inequality follows from our assumption of concavity (4) in Section 2.

If we assume isoelastic utility functions for each user, as in (2), then (13) becomes

$$0 < -c_j \alpha_j x_j^{*\alpha_j-1} + \gamma r_j x_j^{*\gamma-2} < \gamma^2 r_j x_j^{*\gamma-2},$$

where $x_j^* = (r_j/c_j)^{1/(1-\alpha_j-\gamma)}$. Thus, we find the condition

$$0 < -\alpha_j \gamma + \gamma < \gamma^2,$$

or equivalently $0 < 1 - \alpha_j < \gamma$, which we have assumed to hold in Section 2. \square

Thus, the operator can maximize its revenue by offering the lowest possible prices compatible with its resource constraints, i.e., such that the resources required to process all users' jobs do not exceed the resource capacities. We next show that if users have isoelastic utility functions (2), then the amount of each resource required to process users' demanded jobs is a decreasing function of the prices. The resource constraints thus impose a lower bound on the prices offered; in fact, they define a convex set of the prices:

LEMMA 2. *The set $\{\mathbf{p} \mid \mathbf{R}_i \mathbf{x}^*(\mathbf{r}(\mathbf{p})) \leq C_i, i = 1, 2, \dots, n\}$ is convex, where \mathbf{r} is a vector of the per-job price for each user and \mathbf{p} a vector of the prices chosen by the operator. Moreover, $\mathbf{R}_i \mathbf{x}^*(\mathbf{r}(\mathbf{p}))$ is decreasing in each price p_k .*

PROOF. We consider only resource pricing; the result may be similarly proved for bundled or differentiated pricing. It is sufficient to show that x_j^* is a convex function of the prices \mathbf{p} for each user j ; since $\mathbf{R}_i \mathbf{x}^*$ is an affine combination of convex functions, the lemma follows. We first differentiate and find that

$$\frac{\partial x_j^*}{\partial p_k} = \frac{\gamma^{\frac{1}{1-\alpha_j-\gamma}}}{1-\alpha_j-\gamma} \left(\sum_{i=1}^m p_i R_{ij} \right)^{\frac{\alpha_j+\gamma}{1-\alpha_j-\gamma}} R_{kj} < 0$$

since $\gamma > 1 - \alpha_j$ by assumption. Upon differentiating a second time, we find that

$$\frac{\partial^2 x_j^*}{\partial p_k \partial p_l} = \frac{(\alpha_j + \gamma) \gamma^{\frac{1}{1-\alpha_j-\gamma}}}{(1 - \alpha_j - \gamma)^2} \left(\sum_{i=1}^m p_i R_{ij} \right)^{\frac{2\alpha_j+2\gamma-1}{1-\alpha_j-\gamma}} R_{kj} R_{lj}.$$

Thus, the second-derivative matrix of x_j^* equals \mathbf{RQR}^T , where \mathbf{Q} is a diagonal matrix with entries

$$Q_{jj} = \gamma^{\frac{1}{1-\alpha_j-\gamma}} \frac{\alpha_j + \gamma}{(1 - \alpha_j - \gamma)^2} \left(\sum_{i=1}^m p_i R_{ij} \right)^{\frac{2\alpha_j+2\gamma-1}{1-\alpha_j-\gamma}}.$$

A sufficient condition for the second-derivative matrix to be positive semi-definite is then that

$$\gamma^{\frac{1}{1-\alpha_j-\gamma}} \frac{\alpha_j + \gamma}{(1 - \alpha_j - \gamma)^2} \left(\sum_{i=1}^m p_i R_{ij} \right)^{\frac{2\alpha_j+2\gamma-1}{1-\alpha_j-\gamma}} > 0,$$

which holds for each user j . \square

We have thus shown that in order to optimize revenue, the operator should offer the lowest prices possible. However, in doing so the operator could inadvertently favor one user over another, possibly introducing user dissatisfaction. For instance, suppose that one user k requires a relatively large amount of one resource l (e.g., this user runs extremely computationally-intensive jobs and requires a large number of CPU cores). Under resource pricing, the operator might then increase the price p_l of this resource slightly, in order to offer much lower prices for the other resources while keeping the total demand for resource l , $\sum_{j=1}^n R_{lj} x_j^*$ ($\sum_{i=1}^m R_{ij} p_i$), beneath the capacity C_l . But in this scenario, user k would then have a higher per-job cost $\sum_{i=1}^m R_{ik} p_i$ than other users, and thus process a lower number of jobs and receive lower utility $U_j(x_j^*) - r_j x_j^*$. In the next section, we explore ways to limit inequality among users by measuring the fairness resulting from a set of offered prices.

4.2 Fairness

When users submit jobs to the datacenter operator, they are allocated a given amount of each resource, which suffices for the job to be completed. Thus, the operator is performing a *multi-resource allocation* among users, as considered in [6, 11]. In these works, the fairness of a resource allocation is measured in terms of the number of jobs processed for each user—users are assumed to receive utility not from the resources allocated, but rather from the jobs that those resources allowed them to complete. In our framework, since users use utility functions to measure their utility from processed jobs, we measure the fairness of the utility value received (i.e., $\bar{U}_j = U_j(x_j^*) - r_j x_j^*$) by each user j , rather than simply measuring the number of jobs processed for different users.

When measuring the fairness of a distribution, we consider two factors: equitability and efficiency. Equitability refers to the idea that users should receive equitable utility values, e.g., instead of assigning prices so that one user processes a much larger number of jobs and receives a higher utility than another user. However, in some cases one can increase a user's utility level without significantly diminishing other users' utilities: a more unequal allocation can result in more *efficiency*, or a larger amount of total utility

across users. Following [11]'s exploration of this equitability-efficiency tradeoff for multi-resource allocations, we utilize the following fairness functions:

$$\text{sgn}(1 - \beta) \left(\sum_{j=1}^n \bar{U}_j^{1-\beta} \right)^{\frac{1}{\beta}} \left(\sum_{j=1}^n \bar{U}_j \right)^{\lambda+1-\frac{1}{\beta}}, \quad (14)$$

where $\beta \neq 1$ and λ respectively parameterize the equitability and efficiency [12].³ The term $\left(\sum_{j=1}^n \bar{U}_j \right)^{\lambda}$, a function simply of the total utility across users, may be regarded as the efficiency component of this fairness function; as $|\lambda|$ increases, the total utility is more heavily weighted in the function. The remainder of the function, parameterized by a factor β , measures the equitability of the utility distribution. This component is homogeneous in the total utility value, and thus measures only the relative values of the utilities received by each user. We note that if $\lambda = \beta^{-1} - 1$, then maximizing (14) is equivalent to maximizing

$$F_{\beta}(\mathbf{p}) = \frac{1}{1 - \beta} \sum_{j=1}^n \bar{U}_j^{1-\beta}, \quad (15)$$

where $\beta > 0$ parameterizes the type of fairness under consideration. This fairness function corresponds to the well-known α -fairness function with $\alpha = \beta$; we thus observe that as β grows, (15) becomes “more fair” [12].

We next show that when users have isoelastic utility functions, the fairness function (15) becomes a weighted fairness function of the number of jobs processed for each user. To do so, we first note the following lemma:

LEMMA 3. *Given a set of prices \mathbf{p} , then assuming users choose their demand so as to maximize their isoelastic utilities (2),*

$$\begin{aligned} \bar{U}_j &= \left(\frac{\gamma}{c_j} \right)^{\frac{\gamma}{1-\alpha_j-\gamma}} \left(\frac{\gamma}{1-\alpha_j} - 1 \right) r_j^{\frac{1-\alpha_j}{1-\alpha_j-\gamma}} \\ &= \left(\frac{\gamma}{1-\alpha_j} - 1 \right) r_j x_j^{\gamma}. \end{aligned} \quad (16)$$

Thus, the utility received from each user equals the amount paid $r_j x_j^{\gamma}$, multiplied by a price-independent constant.

By choosing the prices to offer, the operator thus determines both the number of jobs processed x_j^* as well as the coefficient $(\gamma/(1 - \alpha_j) - 1) r_j$ of x_j^{γ} in users' total utility. If $\gamma = 1$, this coefficient simply multiplies x_j^* , or the number of jobs processed by user j .

The fairness functions (14) and (15) assume that all users are equal, in the sense that the utility received by each user is treated the same. However, in practice users are heterogeneous—they each submit a distinct type of jobs to the datacenter operator, with different resource requirements and different utility functions. Thus, it may not be equitable to treat all users equally. In Lemma 3, we account for users' differing utility functions with the constant $\gamma/(1 - \alpha_j) - 1$. This constant weight is decreasing in α_j ; thus, as a user becomes less price-sensitive (α_j decreases), the weight in the fairness function increases. More price-sensitive users tend

³These functions are the unique fairness functions satisfying four desirable fairness axioms [12]. In Appendix C, we consider two additional properties (envy-freeness and Pareto-efficiency) that are not axiomatically satisfied.

to receive lower utility values for the same number of jobs due to lower values of α_j in their utility functions (2), an effect that is offset with this weight.

By examining the value of the per-job cost r_j for bundled, resource, and differentiated pricing, we can gain further insight into the interpretation of fairness for each of these pricing plans. Under bundled pricing, we find that $r_j = \mu_j^\gamma p$. Supposing that the ratio of resources in each bundle equals the ratio of resource capacities, we see from (6) that $\mu_j x_j^*$ is simply user j 's dominant share; thus, if there is no volume discount ($\gamma = 1$), then the fairness weight ($\gamma/(1 - \alpha_j) - 1$) r_j is simply the price p multiplied by $1/(1 - \alpha_j) - 1$ and the user's dominant share. Thus, the fairness on users' utility values can be viewed as a (weighted) fairness on the amount of resources required by each user. Users whose jobs have larger resource requirements (larger μ_j) may thus process fewer jobs—their jobs require more of the limited resources, thus detracting from other users' jobs. This *fairness on dominant shares* metric has been previously proposed for general allocations without pricing [6, 11].

If we instead consider differentiated pricing, we note that the weight ($\gamma/(1 - \alpha_j) - 1$) $r_j = (\gamma/(1 - \alpha_j) - 1) \bar{p}_j$ in (16) is completely determined by the operator; users' resource requirements only enter into the choice of \bar{p}_j through the demand functions x_j^* . This pricing scheme thus corresponds to a fairness function that does not as explicitly account for resource requirements.

Under resource pricing with no volume discount, we obtain a fairness metric that lies between the fairness with bundled and differentiated pricing. With resource pricing, we see from (16) that each x_j^* 's weight in the fairness function (14) becomes $(\gamma/(1 - \alpha_j) - 1) \sum_{i=1}^m p_i R_{ij}$. Thus, the number of jobs is multiplied by a constant depending on α_j , multiplied by a weighted sum of a user's resource requirements. Users with higher resource requirements will thus process fewer jobs. However, unlike bundled pricing's dominant shares, the exact weight for each resource is determined by its price, which may be seen as a proxy for how "valuable" different resources are. More valuable resources will have higher prices and thus yield higher weights.

4.3 Fairness-Revenue Tradeoffs

A datacenter operator wishes to optimize both revenue and fairness. In this section, we consider a weighted sum of the revenue (12) introduced in Section 4.1 and the fairness (15) introduced in Section 4.2:

$$\nu \rho(\mathbf{p}) + F_\beta(\mathbf{p}) \quad (17)$$

where $\nu > 0$ is the weight parameter determining the operator's relative emphasis on fairness or revenue. By optimizing a weighted sum of revenue and fairness, the operator can optimize both its revenue and the fairness of users' resulting utility distribution. We consider users with isoelastic utility functions (2) and show that, when the revenue weight ν is sufficiently small, this optimization problem becomes convex. While this result indicates that an operator can more easily optimize fairness, possibly leading to a loss in revenue, we can adapt the proof of Lemma 3 to show that this tradeoff is limited.

We first derive conditions on the weight ν under which maximizing (17) subject to the resource constraints is a convex optimization problem:

Data: Parameters α_j , ν , and γ , tolerance $\epsilon > 0$, update parameter $\mu > 1$.

Result: Optimized resource prices.

Initialize $t \leftarrow 1$;

while $t < m/\text{tol}$ **do**

Find $\mathbf{p}_k^*(t)$ satisfying $f_t(\mathbf{p}) =$

$$-t\nu \nabla \rho(\mathbf{p}) - t \nabla F_\beta(\mathbf{p}) + \sum_{i=1}^m \frac{-\nabla \mathbf{x}^*(\mathbf{p}) \mathbf{R}_i^T}{\mathbf{R}_i \mathbf{x}^*(\sum_{i=1}^m p_i \mathbf{R}_{ij}) - C_i} = 0$$

where k indexes the number of solutions;

Choose k such that $\mathbf{p}_k^*(t)$ minimizes $f_t(\mathbf{p}) =$

$$-t\nu \rho(\mathbf{p}) - t \nabla F_\beta(\mathbf{p}) - \sum_{i=1}^m \log(-\mathbf{R}_i \mathbf{x}^*(\sum_{i=1}^m p_i \mathbf{R}_{ij}) + C_i);$$

Update $\bar{\mathbf{p}} \leftarrow \mathbf{p}_k^*(t)$;

Update $t \leftarrow \mu t$;

end

Algorithm 1: Interior-point optimization of (17).

PROPOSITION 3. Suppose that $\beta > 1$ and that

$$0 \leq \nu \leq \left(\frac{\gamma + \alpha_j - 1}{1 - \alpha_j} \right)^{1-\beta} \gamma^{\frac{\beta\gamma}{\alpha_j + \gamma - 1} - 1} (\beta(1 - \alpha_j) - \gamma) \\ \times \left(\max_i \left(\frac{C_i}{R_{ij}} \right)^{\frac{\beta(\alpha_j - 1)}{\gamma}} \right)$$

for all users j . Then (17) is a concave function of the prices and, by Lemma 2, maximizing (17) subject to the resource constraints is a convex optimization problem.

PROOF. See Appendix B. \square

If ν satisfies Prop. 3's conditions, standard convex optimization solvers may be employed to solve for the optimal prices. Even when ν is larger than the bound given, however, we can adapt these algorithms to find the optimal prices, as shown in Algorithm 1 for the interior-point algorithm. Intuitively, the interior point algorithm starts from a set of prices \mathbf{p} and then uses Lagrange multipliers on the problem constraints to iterate the values of \mathbf{p} so as to increase the objective while remaining within the problem constraints. In our case, differentiating F_β for $\beta > 1$ shows that the objective (17) is decreasing in each price variable—thus, intuitively, the operator searches for the lowest feasible prices. Using this intuition, one can adapt standard proofs of convergence for the interior-point algorithm to show the following:

PROPOSITION 4. Algorithm 1 converges to a price vector \mathbf{p} maximizing the objective (17).

In the case of only one resource, the lowest feasible price for this resource is the price for which the resource constraint is tight; by Lemma 2, since the demands x_j^* are decreasing in the price variables, a unique price exists satisfying this property. Thus, we obtain the following corollary:

COROLLARY 2. Suppose that the operator offers a bundled pricing plan. Then its objective (17) for any revenue weight ν is maximized at the lowest feasible price, which is the zero of the bundled resource constraint $\sum_{j=1}^n p \mu_j^\gamma x_j^*(p)^\gamma = 1$.

We finally note that Algorithm 1 holds only for fixed volume discounts γ . However, if the operator's optimization problem is convex, as in Prop. 3, then the optimal prices for each γ can be determined efficiently. The operator can then perform a line search to determine the optimal volume discount. Section 5.3 numerically characterizes the achieved fairness and revenue for different volume discounts.

The upper bound on the revenue weight ν in Prop. 3 raises some concern that the operator may overly weight fairness

and consequently lose revenue. However, when users have isoelastic utility functions, we can in fact bound the achieved revenue in terms of the achieved fairness when the fairness parameter $\beta > 1$; thus, even if the fairness term in (17) dominates the revenue term, the operator's revenue is still lower-bounded away from zero. Moreover, if $\beta < 1$, a class of fairness measures not included in Prop. 3, then the achieved fairness can be lower-bounded in terms of the achieved revenue. Thus, emphasizing fairness in the operator's objective need not lead to a significant loss of either revenue or of the fairness measured by metrics besides those in Prop. 3.

PROPOSITION 5. *Suppose that $\beta > 1$. The revenue $\rho(\mathbf{p})$ can then be lower-bounded in terms of the achieved fairness (15):*

$$\rho(\mathbf{p}) \geq (F_\beta(\mathbf{p})(1 - \beta))^{\frac{1}{1-\beta}} \sum_{j=1}^n \frac{1 - \alpha_j}{\gamma + \alpha_j - 1}.$$

On the other hand, if $\beta < 1$, then the fairness value $F_\beta(\mathbf{p})$ may be lower-bounded in terms of the achieved revenue:

$$F_\beta(\mathbf{p}) \geq \frac{\rho(\mathbf{p})^{1-\beta}}{1-\beta} \left(\frac{\gamma}{1-\alpha_k} - 1 \right)^{1-\beta},$$

$\alpha_k = \max_j \alpha_j$. Intuitively, a lower value of β more emphasizes efficiency: $\lambda = 1/\beta - 1$ in (14) is increasing in β . By Lemma 3, the efficiency, or total utility received, is simply a scalar multiple of the revenue. A higher β , on the other hand, gives more emphasis on fairness, and thus cannot be lower-bounded by the revenue.

PROOF. See Appendix B. \square

5. NUMERICAL ILLUSTRATIONS

In this section, we use a six-hour workload trace from a Google cluster to illustrate the effects of resource capacity and volume discounts on a datacenter operator's achieved revenue and fairness for different pricing plans [9]. In particular, we find that increasing resource capacity allows improvements in both fairness and revenue, but that these results are highly dependent on the heterogeneity of the users' resource requirements. Varying the volume discount offered allows an increase in fairness, but results in decreased revenue and more leftover resources under resource and differentiated pricing. We find that for a range of resource capacities and volume discounts, all three pricing plans yield comparable revenues, but differentiated pricing permits a significantly higher fairness.

Our workload trace includes 9218 jobs divided into 176580 tasks; each task runs on a single machine within the cluster. The amount of memory (RAM) and fractional number of CPU cores taken up by each active task was recorded at five-minute intervals over 6 hours; these measurements were then passed through a linear transformation to ensure anonymity. For this reason, we omit units in our discussion here. We add each job's resource usage over all the time intervals recorded to find an average per-job CPU usage of 0.136 and memory usage of 0.182.

The jobs in the Google trace show a large variation in resource usage; the CPU and memory usage distributions have respective standard deviations of 13.4 and 18 times their means. To simplify our simulations, we exclude jobs whose total usage of either CPU or memory lies more than one standard deviation away from the mean. We then use

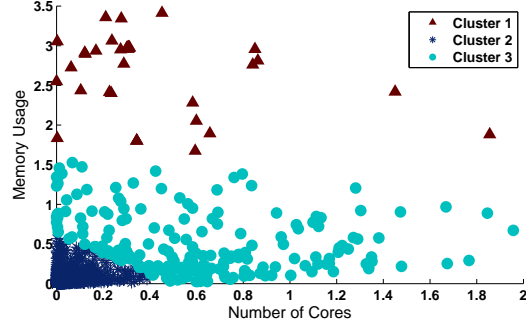


Figure 1: Jobs in the Google trace, clustered based on the total CPU cores and memory used in a 6-hour interval.

k -means clustering to group jobs into three different clusters or types by their CPU and memory usage. Figure 1 shows the resulting clusters of jobs; we ran 30 instances of the k -means algorithm and chose the result minimizing the intra-cluster distance. We take the k -means centroid points to be the resource requirements of each job type: type 1 jobs require 0.4 CPUs and 2.7 units of memory, while type 2 jobs require only 0.01 units of CPU and 0.02 units of memory and type 3 jobs require 0.6 units of CPU and 0.5 units of memory. We associate a user type with each type of job and assume that these users have isoelastic utility functions (2), with parameters $c_j = 1$ and $\alpha_j = 0.4, 0.7, 0.5$ for $j = 1, 2, 3$ respectively. Taking the capacity of each resource to be 6 units, the operator's objective (17) then becomes:

$$\max_{\mathbf{p}} \nu \rho(\mathbf{p}) + F_\beta(\mathbf{p}) \quad (18)$$

$$\text{s.t. } \{0.4x_1^* + 0.01x_2^* + 0.6x_3^*, 2.7x_1^* + 0.02x_2^* + 0.5x_3^*\} \leq 6,$$

where \mathbf{p} is the vector of prices offered for either bundled, resource, or differentiated pricing. For scalability, we suppose that there are 10 users, with 1 user each of types 1 and 3 and 8 users of type 2.

We first consider the effects of resource capacity on the operator's revenue and fairness in Section 5.1, and then show that these tradeoffs can change dramatically with the distribution of users (Section 5.2). Section 5.3 then considers the effects of volume discounts. Throughout our discussion, we take the fairness parameter β in (18) to be relatively large, at $\beta = 20$; this choice of β allows us to approximate max-min fairness, i.e., maximizing the minimum utility value. Larger values of β thus impose a stricter fairness requirement than lower ones, and have been used previously as fairness benchmarks in multi-resource allocation [6, 11].

5.1 Resource Capacity

We first examine the effect of the available capacity on the operator's achieved revenue and fairness. We vary the memory capacity from 1/3 to 8 and use the interior-point algorithm (*cf.* Algorithm 1) to solve for the optimal values of the bundled, resource, and differentiated prices for different revenue weights ν in (18). Figure 2 shows the achieved revenue and fairness for each pricing plan. As the memory capacity increases in Fig. 2, the operator's fairness and efficiency under each pricing plan both improve: the operator has more memory to allocate, and thus more flexibility in setting the prices. Mathematically, we see that one constraint in the optimization problem (18) has been relaxed, allowing the operator to increase the value of its objective

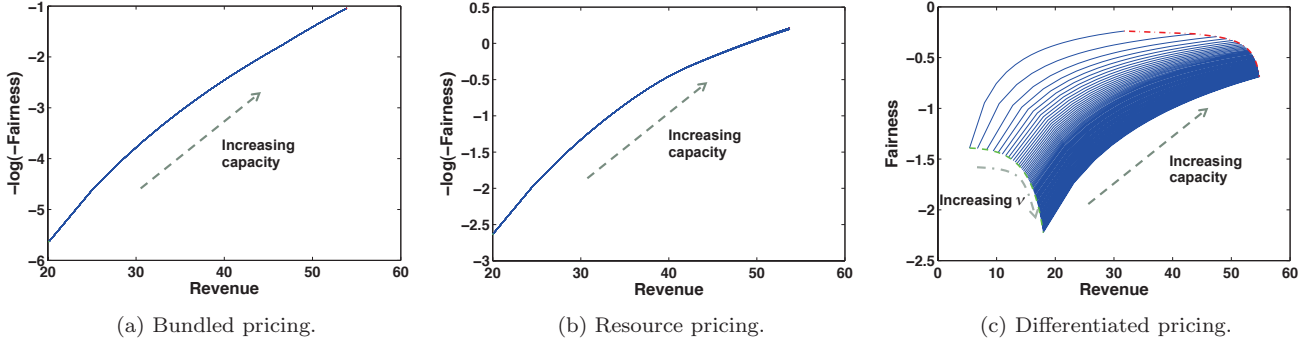


Figure 2: Achieved fairness and revenue for a range of revenue weights ν in (18) and memory capacities. On each contour (individual contours only visible for differentiated pricing), ν is fixed and the capacities are varied. For differentiated pricing, the contours reveal a range of tradeoffs for each fixed capacity and varied weight ν .

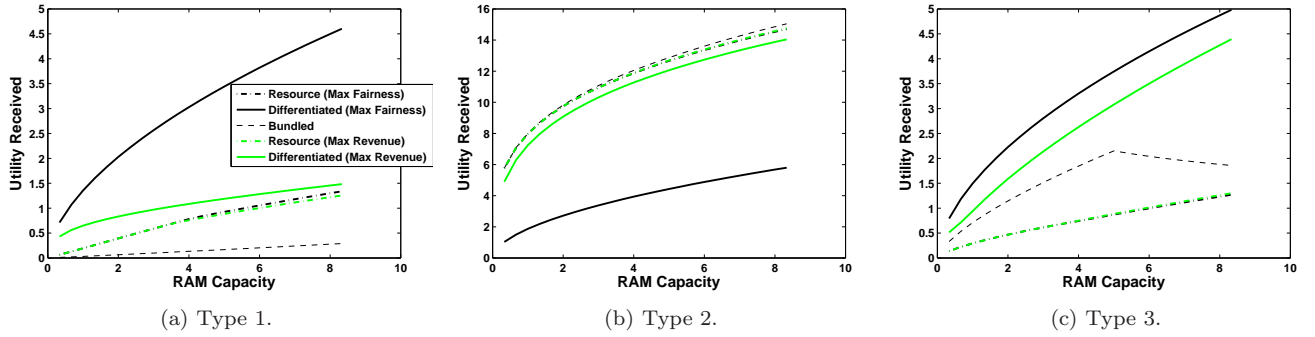


Figure 3: Utility received by the three user types as the RAM (memory) capacity varies for maximum fairness or revenue.

function. We note that, as expected from Corollary 2, the achieved revenue and fairness do not change with ν for bundled pricing. The optimal bundled price, which maximize both revenue and fairness, is simply the lowest price consistent with the operator's resource constraint.

We next compare resource (Fig. 2b) and differentiated (Fig. 2c) pricing. Though differentiated pricing adds only a single additional price variable, it results in a significantly larger fairness for the operator, especially at low memory capacities. Moreover, varying the revenue weight ν results in a much richer set of tradeoffs between fairness and revenue for differentiated, rather than resource, pricing. This result may reflect the large heterogeneity in users' resource requirements: while both type 1 and type 2 users require a very large amount of memory relative to their CPU requirements, type 2 users require significantly less memory and fewer CPUs than type 1 users. If the operator offers resource pricing, type 1 users are then very affected by the memory price, and their utility will be adversely impacted by an increase in this price. In fact, type 1 and type 3 users also require more CPUs per job than type 2 users, so their utility levels will also be adversely impacted by increases in the CPU price. At the same time, type 2 users will process many jobs due to their jobs' relatively low resource requirements cost, allowing the operator to extract more revenue from type 2 users. Under differentiated pricing, the operator can manipulate the prices of individual users to remove these constraints imposed by resource heterogeneity, allowing the operator to increase the fairness across users without a large decreases in revenue.

We can validate the above explanation of the operator's

low achieved fairness for bundled and resource pricing by calculating the utility received by each user under the three pricing plans. Figure 3 shows the utility values for each user when only fairness is optimized (i.e., $\nu = 0$ in (18)), and when only revenue is optimized ($\nu \rightarrow \infty$). We see that type 2 users receive more utility than users of types 1 or 3; since type 2 users have far smaller resource requirements, this result is unsurprising. Users 1 and 3, however, receive almost the same utility under bundled and resource pricing, which is much less than that under differentiated pricing. As the capacity increases, however, their utility levels under resource pricing become visibly larger than those under bundled pricing, corroborating Fig. 2's higher fairness values under resource, rather than bundled, pricing.

5.2 User Heterogeneity

We next examine whether the dramatic changes in fairness for different pricing plans is still observed when users are less heterogeneous. We vary the fraction of type 1 and type 2 users varies (we fix the fraction of type 3 users to be 10%) and fix the memory capacity to be 6 units. Figure 4 shows the resulting fairness and revenue tradeoffs; we see that a visible range of tradeoffs exists for both resource and differentiated pricing, and that the fairness for different revenue weights ν is much less negative than in Fig. 2. As the fraction of type 1 users increases, the revenue decreases under all pricing plans: type 1 users require more resources than type 2 users, so the operator receives more revenue from using its available resources to process type 2 users' jobs. Under bundled pricing, the fairness also increases with the fraction of type 1 users; all type 1 users receive the same (lower than

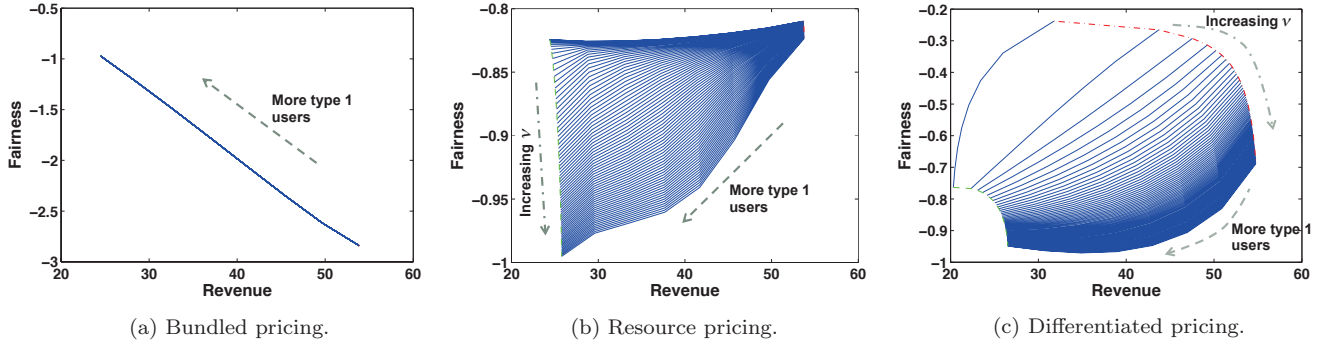


Figure 4: Achieved fairness and revenue for a range of revenue weights ν and distribution of user types. On each contour (individual contours not visible in (a)), the weight ν is fixed and the fraction of type 1 users varies. For (b) resource and (c) differentiated pricing, the contours reveal a range of tradeoffs for each distribution of user types and varied revenue weights ν .

type 2) utility values, increasing equitability. Under resource or differentiated pricing, however, increasing the fraction of type 1 users decreases the fairness. To achieve a higher revenue, the operator processes more jobs from type 2 users, even when more type 1 users are present.

As expected from Corollary 2, varying the revenue weight ν has no effect on the utility or revenue from bundled pricing. Under resource or differentiated pricing, however, as the revenue weight ν increases in Fig. 4, the operator’s revenue increases for any given fraction of type 1 users. However, this increase in revenue results in a decrease in fairness: in particular, when a low fraction of users is present in resource pricing (e.g., the leftmost contour in Fig. 4b), the revenue barely increases, while the fairness drops steeply as ν increases. We can explain this result by examining the utility received by each user for different fractions of type 1 users, as shown in Fig. 5.

We see from Fig. 5 that as the fraction of type 1 users increases, the minimum utility over all three users also decreases, though type 2 and type 3 users actually receive more utility under bundled and resource pricing when the revenue is maximized. This is offset, however, by a decrease in utility for type 1 users, which results in a decrease in the minimum utility across users. This result likely comes from the fact that type 1 users require more resources to process their increased number of jobs, as the fraction of type 1 users increases. For bundled pricing, however, the minimum utility increases when either revenue or fairness is maximized, likely because type 1 users, who had previously received lower utilities than type 2 or 3 users, could now process more jobs. When fewer type 1 users are present, the price of type 1 users’ jobs under resource or differentiated pricing may be set so that these users receive slightly higher utility values; since there are relatively few type 1 users, they do not take away many resources from the other types of users.

5.3 Volume Discounts

We conclude by examining a parameter set by the operator: the volume discount γ . Figure 6 shows the fairness-revenue tradeoffs for a range of volume discounts γ .

We see that for both bundled and resource pricing, varying ν in the operator’s objective function (18) does not yield an appreciable tradeoff (for bundled pricing, this result is again Corollary 2). Moreover, as γ decreases, the fairness decreases quite rapidly under both these pricing plans; once

the revenue exceeds 50, the fairness value decreases exponentially. Differentiated pricing allows a wider range of much higher fairness values. As the revenue weight ν increases from 0, the revenue under differentiated pricing increases rapidly, while the range of achieved fairness stays roughly the same; though each contour represents a linear increase in ν , the resulting increase in revenue, measured by the “distance” between the contours, is very pronounced. As ν increases further, the contours grow closer together, but the range of achieved fairness decreases more rapidly.

For larger revenue weights ν , we find that the operator’s revenue decreases with γ for all three pricing plans, as a result of decreased demand from users. This demand decrease, however, affects users in different ways and acts to increase fairness. When $\nu = 0$ under differentiated pricing, the revenue does not decrease monotonically with γ ; for a small volume discount (γ near 1), the revenue initially increases and then decreases with γ . Unlike our results for varied memory capacities or distributions of users (Figs. 2 and 4), the differentiated pricing contours for different ν actually overlap: the operator can achieve the same combination of fairness and revenue by optimizing two different weighted sums of fairness and revenue while offering different volume discounts.

We can explain the very negative fairness values observed in Figs. 6a and 6b for bundled and resource pricing by examining the equitability and efficiency components of the fairness function (15). We see that the equitability is quite negative, even for differentiated pricing, where the fairness values are relatively moderate (Fig. 6c). Thus, the very low fairness values observed in Fig. 6 may be attributed more to lack of equitability in the utilities received by different users rather than a low total utility.

The relatively low efficiency values observed in Fig. 7 lead us to consider another measure of efficiency: leftover resource capacity. Though leftover capacity is not explicitly accounted for in the user’s optimization problem (18)—capacity only enters through the resource constraints—the leftover capacity may indicate the potential additional revenue that an operator could achieve, e.g., if users’ demand functions change due to different volume discounts offered. We find that for all volume discounts γ and weights ν , only CPU cores are leftover; as the revenue increases, the leftover capacity increases for both resource and differentiated pricing. Thus, a volume discount reduces the “wasted” capacity

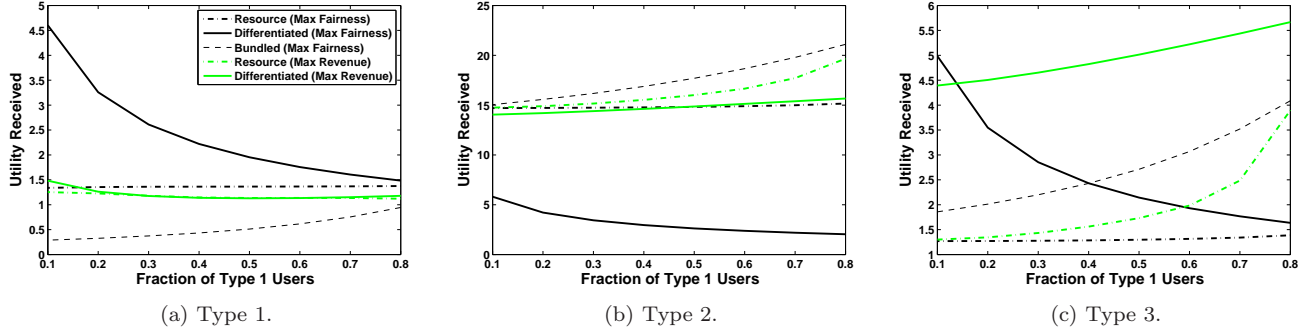


Figure 5: Utility received by the three user types as the fraction of type 1 and 2 users varies, maximum fairness or revenue.

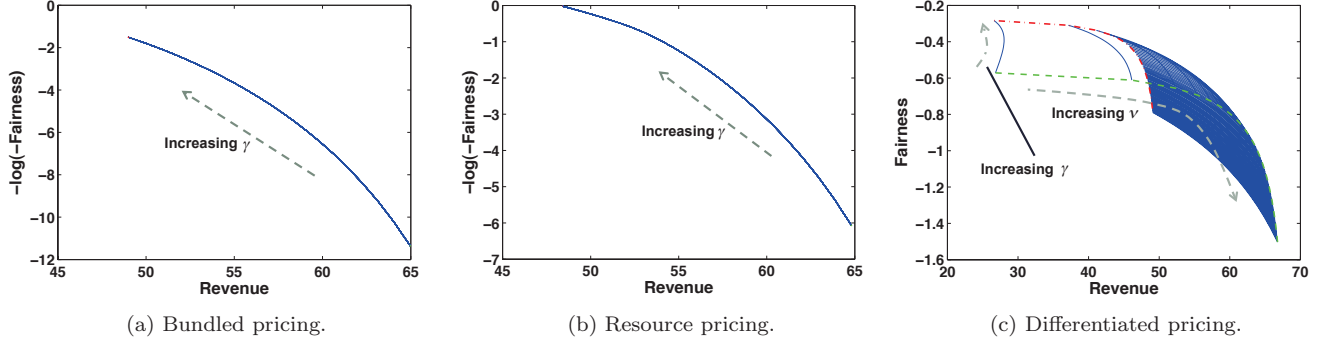


Figure 6: Achieved fairness and revenue for a range of revenue weights ν and volume discounts γ . On each contour (individual contours only visible for differentiated pricing), ν is fixed and γ varies. For (a) bundled and (b) resource pricing, the optimal fairness and revenue values do not change significantly with ν . For (c) differentiated pricing, the contours reveal a range of tradeoffs for each fixed γ and varied ν .

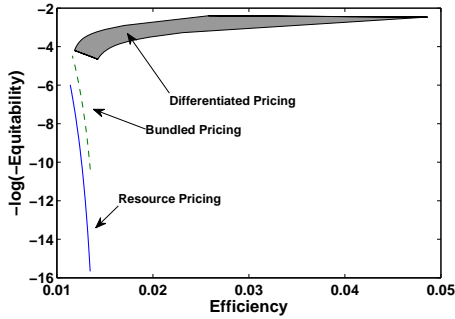


Figure 7: Equitability and efficiency components of the fairness function (15) as the revenue weight ν varies in the operator's objective function (18).

by increasing user demand and operator revenue. For bundled pricing, however, we have the opposite scenario: as the revenue and overall user demand increase, the leftover capacity also increases. With bundled pricing, users are forced to purchase excess capacity according to the ratio of resources in the resource bundle. As the volume discount increases, user demand goes up and users purchase more excess capacity, which is included in the leftover capacity shown in Fig. 7. This result thus highlights the “inefficiency” inherent in forcing users to pay for resources they do not need.

6. CONCLUSION AND FUTURE WORK

In this work, we develop a mathematical framework for evaluating three cloud pricing strategies—bundled, resource,

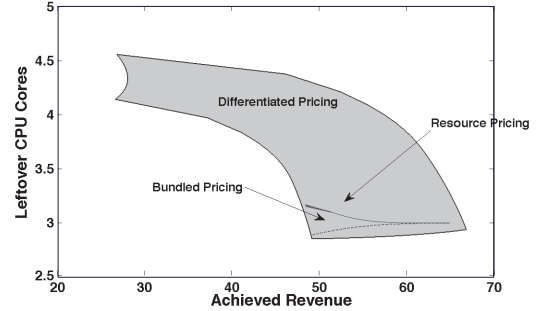


Figure 8: Leftover CPU capacity versus achieved revenue as the volume discount γ and revenue weight ν vary in the operator's objective function (18).

and differentiated pricing—in terms of their resulting fairness and revenue. We first characterize user demand for resources as a function of the prices offered under these different pricing plans, and then find the resulting revenue. We next formulate a family of metrics to measure the fairness of the offered prices and analyze the tradeoff between fairness and revenue. After showing some analytical bounds on this tradeoff, we finally use data taken from a Google cluster to numerically evaluate the impact of resource capacity and volume discounts on the operator's fairness-revenue tradeoff. We find that differentiated pricing, even if it only adds one additional price variable compared to resource pricing, can significantly increase the fairness under bundled or resource pricing, though revenue remains comparable across

all pricing strategies. The operator can also increase both fairness and revenue through an increase in resource capacity, but the magnitude of this benefit depends heavily on the heterogeneity of users' resource requirements. Finally, a lower volume discount can be used to increase fairness, but at a loss of some revenue and increase in unused resource capacity.

Throughout this work, we consider fairness and revenue solely in terms of prices. A real implementation, however, would allow us to explore a different aspect of fairness: user priorities. Instead of choosing prices so as to balance fairness and revenue, an operator could instead leverage the temporal dimension of job processing in datacenters to orthogonalize an operator's desire for fairness with its incentive to increase revenue. Users receiving lower utility values could be compensated with higher priorities, allowing their jobs to be completed faster. By implementing such a prioritized allocation scheme and devising algorithms for computing these priorities, we can evaluate whether setting user priorities, choosing job prices under one of many possible pricing plans, or a combination of both is most effective at balancing a datacenter operator's competing objectives of maximizing revenue and fairness.

7. REFERENCES

- [1] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafir. The resource-as-a-service (RaaS) cloud. In *Proc. of USENIX HotCloud*, pages 12–12. USENIX Association, 2012.
- [2] Amazon. EC2 Pricing, November 2012. <http://aws.amazon.com/ec2/pricing/>.
- [3] L. Du. Pricing and resource allocation in a cloud computing market. In *Proc. of IEEE/ACM CCGrid*, pages 817–822. IEEE, 2012.
- [4] D. Durkee. Why cloud computing will never be free. *Queue*, 8(4):20, 2010.
- [5] G. Feng, S. Garg, R. Buyya, and W. Li. Revenue maximization using adaptive resource provisioning in cloud computing environments. In *Proc. of the ACM/IEEE Conf. on Grid Computing*, pages 192–200. IEEE, 2012.
- [6] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Proc. of USENIX NSDI*, pages 24–37. USENIX, 2011.
- [7] Google. Compute Engine Pricing, November 2012. <https://cloud.google.com/pricing/compute-engine>.
- [8] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel. The cost of a cloud: Research problems in data center networks. *ACM SIGCOMM Computer Communication Review*, 39(1):68–73, 2008.
- [9] J. L. Hellerstein. Google cluster data. Google research blog, Jan 2010. <http://googleresearch.blogspot.com/2010/01/google-cluster-data.html>.
- [10] T. Henzinger, A. Singh, V. Singh, T. Wies, and D. Zufferey. FlexPRICE: Flexible provisioning of resources in a cloud environment. In *Proc. of IEEE CLOUD*, pages 83–90. IEEE, 2010.
- [11] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang. Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework. In *Proc. of IEEE INFOCOM*, pages 1206–1214. IEEE, 2012.
- [12] T. Lan, D. Kao, M. Chiang, and A. Sabharwal. An axiomatic theory of fairness in network resource allocation. In *Proc. of IEEE INFOCOM*, pages 1–9. IEEE, 2010.
- [13] I. Menache, A. Ozdaglar, and N. Shimkin. Socially optimal pricing of cloud computing resources. In *Proc. of the Intl. ICST Conf. on Performance Evaluation Methodologies and Tools*, pages 322–331. ICST, 2011.
- [14] Microsoft. Windows Azure Pricing Calculator, November 2012. <http://www.windowsazure.com/en-us/pricing/calculator/?scenario=cloud>.
- [15] W. Nicholson and C. Snyder. *Microeconomic Theory*. South-Western Cengage Learning, 10th edition, 2008.
- [16] A. Odlyzko. Network neutrality, search neutrality, and the never-ending conflict between efficiency and fairness in markets. *Review of Network Economics*, 8(1):40–60, 2009.
- [17] P. Samimi and A. Patel. Review of pricing models for grid & cloud computing. In *Proc. of IEEE ISCI*, pages 634–639. IEEE, 2011.
- [18] E. Siham, C. Schlereth, and B. Skiera. Price comparison for infrastructure-as-a-service. In *Proc. of the Eur. Conf. on Information Systems*. AIS, 2012.
- [19] F. Teng and F. Magoules. Resource pricing and equilibrium allocation policy in cloud computing. In *Proc. of IEEE CIT*, pages 195–202. IEEE, 2010.
- [20] Y. Teo and M. Mihailescu. A strategy-proof pricing scheme for multiple resource type allocations. In *Proc. of IEEE ICPP*, pages 172–179. IEEE, 2009.
- [21] K. Tsakalozos, H. Killapi, E. Sitaridi, M. Roussopoulos, D. Paparas, and A. Delis. Flexible use of cloud resources through profit maximization and price discrimination. In *Proc. of the IEEE Intl. Conf. on Data Engineering (ICDE)*, pages 75–86. IEEE, 2011.
- [22] S. Wagner, E. van den Berg, J. Giacomelli, A. Ghetie, J. Burns, M. Tauil, S. Sen, M. Wang, M. Chiang, T. Lan, R. Laddaga, P. Robertson, and P. Manghwani. Autonomous, collaborative control for resilient cyber defense (ACCORD). In *Workshop on Adaptive Host and Network Security*, 2012.
- [23] H. Wang, Q. Jing, R. Chen, B. He, Z. Qian, and L. Zhou. Distributed systems meet economics: Pricing in the cloud. In *Proc. of USENIX HotCloud*, pages 6–6. USENIX Association, 2010.
- [24] W. Wang, P. Zhang, T. Lan, and V. Aggarwal. Datacenter net profit optimization with deadline dependent pricing. In *Proc. of CISS*, pages 1–6. IEEE, 2012.
- [25] S. Wee. Debunking real-time pricing in cloud computing. In *Proc. of IEEE/ACM CCGrid*, pages 585–590. IEEE, 2011.
- [26] C. Weinhardt, A. Anandasivam, B. Blau, N. Borissov, T. Meinel, W. Michalk, and J. Stöcker. Cloud computing—a classification, business models, and research directions. *Business & Information Systems Engineering*, 1(5):391–399, 2009.
- [27] H. Xu and B. Li. Maximizing revenue with dynamic cloud pricing: The infinite horizon case. In *Proc. of IEEE ICC, Next-Generation Networking Symposium*. IEEE, 2012.

APPENDIX

A. INTRODUCING DEADLINES

In this section, we show that our formulation in Sections 2 - 4 may be altered to allow changes in user utilities and job demands over multiple time intervals. In formulating this problem, we account for varying job deadlines: users specify a completion deadline when a job is submitted, and the operator takes these into account when setting prices for each time interval. We suppose that the operator sets different prices in each time interval; then if the price optimization for the single interval is convex (e.g., as in Prop. 3), so is the optimization over prices in all time intervals. We derive this extension only for resource pricing, but similar methods can be used for bundled or differentiated pricing.

We suppose that when users submit jobs to the datacenter operator in each time interval, they specify the maximum time to complete these jobs as well as the job resource requirements that, if met, will ensure that the job is completed before the deadline. We suppose that it does not matter when a user receives the requested resources, so long as the resources are allocated in the requested ratio. For instance, a job with a deadline of three time intervals and a requirement of 6 CPU slots and 3 GB of RAM might be allocated 2 slots and 1 GB in each time interval; or 2 slots and 1 GB in the first, no resources in the second, and 4 slots and 2 GB in the third time interval; either allocation allows the job to complete before the deadline. In each time interval, each user submits one type of job to the datacenter (i.e., R_{ij} is the same for all jobs of user j and all resources i submitted during a given time interval). If a user has multiple types of jobs to run, these may be treated as coming from distinct users. One could also group users with similar per-job resource requirements.

We consider a finite time horizon of T time intervals, indexed by $s = 1, 2, \dots, T$, and assume that all job deadlines fall before the final time T . At the beginning of each time interval s , the operator sets the resource prices $p_{i,s}$ for each resource i ; denoting the vector of resource prices in period s by \mathbf{p}_s , each user j then submits $x_{j,s}^*(\mathbf{p}_s)$ number of jobs to the operator, with associated deadline $\tau_{j,s}$. These jobs are charged according to the price \mathbf{p}_s , though the resource prices may change before the job completes. Users choose the number of jobs so as to maximize their utility in period s , and do not account for any jobs that have been already submitted or may be submitted in future periods. For simplicity, we suppose that for each user j , all of user j 's jobs submitted in a given period s have the same deadline.

We now introduce the variables $x_{j,s}(t)$, $s < t$, to denote the number of jobs submitted by user j in period s that are processed during time interval t . To ensure that all deadlines are met, we introduce the constraints

$$\sum_{t=s}^{\tau_{j,s}} x_{j,s}(t) \geq x_{j,s}^*(\mathbf{p}_s) \quad (19)$$

for each user j and time interval s ; since $x_{j,s}^*$ is convex in the prices \mathbf{p}_s by Lemma 2, this constraint defines a convex set. The resource constraints for each period t then become

$$\sum_{s=1}^t \sum_{j=1}^n R_{ij}(s) x_{j,s}(t) \leq C_i, \quad (20)$$

where $R_{ij}(s)$ denotes the per-job requirement of resource i

for jobs submitted by user j in the time interval s . We note that these constraints are affine in the variables $x_{j,s}(t)$; thus, the constraints (19) and (20) together define a convex set in the variables \mathbf{p}_s and $x_{j,s}(t)$. We then find that the revenue with deadlines may be expressed as

$$\sum_{s=1}^T \left(\sum_{j=1}^n x_{j,s}^* \gamma \sum_{i=1}^m R_{ij}(s) p_{i,s} \right). \quad (21)$$

The fairness may be thought of as the sum of the fairness of the utilities received in each time interval; the fairness function is thus only indirectly dependent on the deadlines $\tau_{j,s}$. Users specify the deadlines independent of the prices offered, so these should not affect their perceived utility or fairness. Thus, (15) yields the fairness function

$$\frac{1}{1-\beta} \sum_{s=1}^T \left(\sum_{j=1}^n \bar{U}_j(x_{j,s}^*(\mathbf{p}_s), \mathbf{p}_s)^{1-\beta} \right). \quad (22)$$

We note that since both the revenue (21) and fairness (22) are simply the revenue and fairness in each time interval s added together, we may apply Prop. 3 to formulate a weighted sum (with weights possibly depending on the time interval) of both the revenue and fairness that is a concave function. We then have the following result:

PROPOSITION 6. *The optimization problem*

$$\begin{aligned} \max_{x_{j,s}(t), p_{i,s}} \quad & \sum_{s=1}^T \nu(s) \left(\sum_{j=1}^n x_{j,s}^* \gamma \sum_{i=1}^m R_{ij}(s) p_{i,s} \right) \\ & + \frac{1}{1-\beta} \left(\sum_{j=1}^n \bar{U}_j(x_{j,s}^*(\mathbf{p}_s), \mathbf{p}_s)^{1-\beta} \right) \\ \text{s.t.} \quad & \sum_{s=1}^t \sum_{j=1}^n R_{ij}(s) x_{j,s}(t) \leq C_i; \quad 1 \leq t \leq T; \quad 1 \leq i \leq m \end{aligned} \quad (23)$$

$$\sum_{t=s}^{\tau_{j,s}} x_{j,s}(t) \geq x_{j,s}^*(\mathbf{p}_s); \quad 1 \leq s \leq T; \quad 1 \leq j \leq n \quad (24)$$

is a convex optimization if, for each time interval s , the weight $\nu(s)$ is chosen so as to satisfy Prop. 3 for the revenue and fairness in period s .

B. PROOFS

B.1 Proposition 3

PROOF. As in the proof of Lemma 2, we show the result for resource pricing; the proofs for bundled or differentiated pricing are analogous. We first note that the second derivative matrix of (17) may be written as

$$\nu \nabla^2 \rho + \nabla^2 \mathbf{F}_\beta = [\mathbf{R}_{ij}^\gamma]^\top \mathbf{Q} [\mathbf{R}_{ij}^\gamma],$$

where \mathbf{Q} is a diagonal matrix with entries

$$\begin{aligned} Q_{jj} = & \nu(\gamma r_j(\mathbf{p}))^{\frac{2\gamma + \alpha_j - 1}{1 - \alpha_j - \gamma}} \gamma^2 \frac{1 - \alpha_j}{(1 - \alpha_j - \gamma)^2} \\ & + \gamma^{\frac{(1-\beta)\gamma}{1-\alpha_j-\gamma}} \left(\frac{\gamma + \alpha_j - 1}{1 - \alpha_j} \right)^{-\beta} \frac{\beta(1 - \alpha_j) - \gamma}{1 - \alpha_j - \gamma} \\ & \times r_j^{\frac{(-1-\beta)(1-\alpha_j)+2\gamma}{1-\alpha_j-\gamma}} \end{aligned}$$

and $[\mathbf{R}_{ij}^\gamma]$ is an $m \times n$ matrix with (i, j) entry R_{ij}^γ . A sufficient condition for this second-derivative matrix to be negative-semidefinite is that $Q_{jj} < 0$ for each j . Thus, we find the condition

$$\nu \leq \left(\frac{\gamma + \alpha_j - 1}{1 - \alpha_j} \right)^{1-\beta} \gamma^{\frac{\beta\gamma}{\alpha_j + \gamma - 1} - 1} (\beta(1 - \alpha_j) - \gamma) \times (r_j)^{\frac{\beta(1 - \alpha_j)}{\alpha_j + \gamma - 1}}.$$

. We can remove the price variables from this bound by observing that the resource constraints $\mathbf{R}\mathbf{x}^* \leq \mathbf{C}$ imply that

$$R_{ij} r_j^{\frac{\gamma}{1 - \alpha_j - \gamma}} \leq C_i$$

for each resource i and user j ; thus,

$$r_j^{\frac{\beta(1 - \alpha_j)}{\alpha_j + \gamma - 1}} \geq \max_i \left(\frac{C_i}{R_{ij}} \right)^{\frac{\beta(\alpha_j - 1)}{\gamma}}.$$

We therefore find the sufficient condition

$$\nu \leq \left(\frac{\gamma + \alpha_j - 1}{1 - \alpha_j} \right)^{1-\beta} \gamma^{\frac{\beta\gamma}{\alpha_j + \gamma - 1} - 1} (\beta(1 - \alpha_j) - \gamma) \times \left(\max_i \left(\frac{C_i}{R_{ij}} \right)^{\frac{\beta(\alpha_j - 1)}{\gamma}} \right).$$

□

B.2 Proposition 5

PROOF. Suppose that $\beta < 1$ in (15). We first introduce the notation

$$\nu_j = \frac{\gamma + \alpha_j - 1}{1 - \alpha_j} = \frac{\gamma}{1 - \alpha_j} - 1,$$

and find that the fairness may be written as

$$\begin{aligned} & \frac{1}{1 - \beta} \sum_{j=1}^n \nu_j^{1-\beta} \left(\frac{\gamma}{c_j} \right)^{\frac{\gamma(1-\beta)}{1-\alpha_j-\gamma}} r_j^{\frac{(1-\alpha_j)(1-\beta)}{1-\alpha_j-\gamma}} \\ & \geq \frac{\min_j \nu_j^{1-\beta}}{1 - \beta} \left(\sum_{j=1}^n \left(\frac{\gamma}{c_j} \right)^{\frac{\gamma}{1-\alpha_j-\gamma}} r_j^{\frac{1-\alpha_j}{1-\alpha_j-\gamma}} \right)^{1-\beta} \\ & = \frac{\rho(\mathbf{p})^{1-\beta}}{1 - \beta} \left(\frac{\gamma}{1 - \alpha_k} - 1 \right)^{1-\beta}. \end{aligned}$$

The first inequality uses the sub-additivity of the function $f(x) = x^{1-\beta}$, and we let k be such that $\alpha_k = \min_j \alpha_j$.

We now suppose that $\beta > 1$ and find that for each user j ,

$$\nu_j^{1-\beta} \left(\frac{\gamma}{c_j} \right)^{\frac{\gamma(1-\beta)}{1-\alpha_j-\gamma}} r_j^{\frac{(1-\alpha_j)(1-\beta)}{1-\alpha_j-\gamma}} \leq F_\beta(\mathbf{p})(1 - \beta),$$

which is equivalent to

$$\left(\frac{\gamma}{c_j} \right)^{\frac{\gamma}{1-\alpha_j-\gamma}} r_j^{\frac{1-\alpha_j}{1-\alpha_j-\gamma}} \geq \nu_j^{-1} (F_\beta(\mathbf{p})(1 - \beta))^{\frac{1}{1-\beta}}.$$

Adding these lower bounds and using the definition of ν_j , we find that

$$\rho(\mathbf{p}) \geq (F_\beta(\mathbf{p})(1 - \beta))^{\frac{1}{1-\beta}} \sum_{j=1}^n \frac{1 - \alpha_j}{\gamma + \alpha_j - 1}.$$

□

C. FAIRNESS PROPERTIES

While the fairness functions (14) are the unique functions satisfying certain desirable fairness axioms, other desirable fairness properties are not axiomatically satisfied by these functions. In this appendix, we consider two such properties: envy-freeness and Pareto-eficiency.

Definition 3. **Envy-freeness** holds if and only if no user envies another user's allocation. Mathematically, let r_{ij} denote the amount of resource i received by user j . User j can then process $\max_i r_{ij}/R_{ij}$ number of jobs. Envy-freeness is defined as the property that $\max_i r_{ij}/R_{ij} > \max_i r_{ik}/R_{ik}$ for any $j \neq k$. In words, no other user's allocation would enable a user to process more jobs than her allocation would.

Definition 4. A function f is **Pareto-efficient** if, whenever \mathbf{x} Pareto-dominates \mathbf{y} (i.e., $x_i \geq y_i$ for each index i and $x_j > y_j$ for some j), $f(\mathbf{x}) > f(\mathbf{y})$.

We first show that envy-freeness always holds for an allocation of utilities if users optimize the number of jobs that they submit under bundled or resource pricing:

PROPOSITION 7. *Suppose that each user chooses a number of jobs to submit, $x_j^*(r_j(\mathbf{p}))$, so as to maximize her utility function. Then for any given set of resource or bundled prices \mathbf{p} , if two users switch their corresponding resource allocations and are required to pay the difference in resources received, neither gains any utility. If users do not need to pay this difference, then a user may gain utility by switching allocations.*

PROOF. The first part of the proposition is trivial, as for a given set of prices, users are assumed to optimize the number of jobs processed. Since all users are charged at the same per-resource or per-bundle rates, if switching resource allocations alters the number of jobs processed for either, neither gains any utility. If switching allocations results in each user being able to process the same number of jobs, then the users have either not changed their resource allocations or have gained resources; in the latter case, their utility decreases due to paying for extra resources that they do not use to process jobs.

If users are not required to pay the difference in prices from switching resource allocations, then one user may gain utility by switching resource allocations with another: consider, for instance, a user who processes a very small amount of jobs, due to a low willingness to pay (e.g., a high value of α_j and low value of c_j in (2)'s utility functions). Then this user will process a higher number of jobs by switching resources allocations with another user who is more willing to pay for resources in exchange for processing jobs. □

We next find that the fairness function (14) is Pareto-efficient if $\beta > 0$ and $|\lambda| \geq 1/\beta - 1$ [11, 12]. Thus, if the allocation $\bar{U}_j(x_j^*, \mathbf{p})$ Pareto-dominates the allocation $\bar{U}_j(x_j^*, \mathbf{q})$, then the fairness function (15), for which $\lambda = 1/\beta - 1$, attains a larger value under the prices \mathbf{p} than it does under the prices \mathbf{q} . The prices here may represent those under any pricing plan; bundled, resource, or differentiated. If Lemma 3 holds and the utility received from each user j is equal to a price-independent constant multiplied by the amount paid by user j , then the revenue (9) is also Pareto-efficient, as it is an affine transformation of the sum of the utilities received. This latter quantity corresponds to the fairness (14) as $\lambda \rightarrow \infty$.